

REMARKS

Claims 1-24 are pending in the present application. Claims 1, 4, 12, 15, 23, 24 are amended. Claims 1, 12, and 23 are amended to recite "wherein the step of performing data conversion between the object reference and the client object for different types of data comprises: responsive to receiving a method called by the client object, parsing the method for an argument; performing data conversion on the argument if necessary; determining if the argument is wrapped by an adapter; if the argument is wrapped by an adapter, unwrapping the adapter from the argument to retrieve an object reference of the argument, wherein the object reference of the argument provides a reference to a server object in the second programming environment representing the argument; delegating the method called by the client object to the object reference of the argument; responsive to detecting a return value, determining if the return value is an object reference for the second programming environment; if the return value is an object reference for the second programming environment, wrapping the object reference with a suitable adapter based on a type of the object reference; and returning the wrapped object reference to the client object." These features are supported at least on page 34, line 1 to page 37, line 10 and in Figures 10A-C and 11 of the current specification.

Claims 4, 15, and 24 are amended to recite "wherein the step of performing data conversion between the proxy and the client object for different types of data comprises: responsive to receiving a method called by the client object, parsing the method for an argument; performing data conversion on the argument if necessary; determining if the argument is wrapped by an adapter; if the argument is wrapped by an adapter, unwrapping the adapter from the argument to retrieve a proxy of the argument, wherein the proxy of the argument provides a reference to a server object in the second programming environment representing the argument; delegating the method called by the client object to the proxy of the argument; responsive to detecting a return value, determining if the return value is a proxy for the second programming environment; if the return value is proxy for the second programming environment, wrapping the proxy with a suitable adapter based on a type of the object reference; and returning the wrapped proxy to the client object." These features are supported at least on page 34, line 1 to page 37, line 10 and in Figures 10A-C and 11 of the current specification.

No new matter is added as a result of the above amendments. Reconsideration in view of the above amendment to claims in view of the following Remarks is respectfully requested.

I. 35 U.S.C. § 102(a), Alleged Anticipation, Claims 1-4, 7-9, 11-15, and 18-24

The Office Action rejects claims 1-4, 7-9, 11-15, and 18-24 under 35 U.S.C. § 102(a) as being allegedly anticipated by Trevor et al., "The Use of Adapters to Support Cooperative Sharing", 1994 ACM, pages 219-230. This rejection is respectfully traversed.

As to claims 1, 14, 12, 15, 23 and 24, the Office Action states:

As to claims 1,4,12,15,23 and 24, Trevor teaches a process for invoking a method of a server object (i.e., *"Interfaces describe the means by which a client interacts with the services provided by the object."* The preceding text excerpts clearly indicate that the client interact with a server object through an interface." (page 222, col 1) in a distributed application in a distributed data processing system (page 219, col 1), the process comprising the computer-implemented steps of executing a client object (i.e., *"Dave et al. [7] extend the initial definition of proxy objects [22], as local representation of remote objects"* The preceding text excerpts clearly indicate that the client object or the proxy object is the local representation of the server/target object. It is thru a proxy object that a client application interacts with the server/target object.) (page 221, col 2) in a client that implements a first programming environment (i.e., *"In supporting co-operation among a wide set of different users and platforms, possibly with different interfaces to applications and even diverse language bindings, a uniform and clean event mechanism is needed."* The preceding text excerpts clearly indicate that the client binds to a server object, which is of different language. Therefore a client object of language 1 interacts with server object of language 2.) (page 224, col 1), said client object (page 221, col 2) being written for said first programming environment (page 224, col 1); executing a server object (page 222, col 1) in a server that implements a second programming environment (page 224, col 1) that is different from said first programming environment (page 224, col 1), said server object being written for said first programming environment (page 224, col 1); executing (i.e., *"A client may use any adapter or object within the service, but to do so they must first request that the service bind them to the required object or adapter."* ... *"Successful completion of the binding process on an adapter object forms an invocation path between the client and the underlying object (s)."* The preceding text excerpts clearly indicate that the client requests the services from the underlying server objects. The services are requested only thru invocation of object

methods.) (page 226, col 1) said client object (page 221, col 2) which begins an attempt to invoke a method (page 226, col 1) in the server object; in response to the client object beginning the attempt to invoke the method in the server object (page 222, col 1): obtaining an object reference (i.e., *"In order to create an object, a client passes an object template to the factory. The template is tested against the factories template list. If a successful match is made, the object, stored along the matched template, is created from its definition, and the resulting reference to the new object is returned to the client."* The preceding text excerpts clearly indicate that the client obtains an object reference/ client proxy by sending its template e.g. name to the factory.) (page 225, col 2) on the client (page 225, col 2) for said second programming environment (page 224, col 1); wrapping the object reference in an adapter (i.e., *"Interfaces describe the means by which a client interacts with the services provided by the object. Interface adapters provide facilities that abstract over these services to provide different services to users based upon context."* ... *"Adapters provide a level of indirection between object interfaces and object users. This level of indirection allows interface adapters to provide additional facilities to manage the invocation of methods depending upon the context within which they are defined."* ... *"At no point is an underlying object interface directly visible to a client, and the only way of invoking the operations is through an object adapter."* ... *"Adapter A, Interface I adapts Object B, with Interface J This indicates that the adapter object A is an adapting object over object B, which may be a simple object or an adapter itself."* The preceding text excerpt clearly indicate that the adapter adapts the object reference of the server or the adapted. The definition of an adapter can be found in the design pattern book of Gamma. The adapter by its definition adapts/ wraps the target object reference and thus provides the client of the adapter the services provided by the adapted object e.g., adapter A provides the services of adapter object B) (page 222, col 1, col 2; page 225, col 2) which isolates the object reference (i.e., the object B) (page 222, col 1, col 2; page 225, col 2) from said client object and the adapter performing data conversion (i.e., *"The basic structure of object adapters is as a set of mappings which map method names from those presented to users to those used within the object interface."* The preceding text excerpts clearly indicate that the adapter maps/converts between the adapter interface and the actual server object interface. Therefore, a client's invocation of an adapter method actually invokes a mapped server object method.) (page 222, col 2) between the object reference and the client object (page 221, col 2) for different types of data (page 224, col 1); calling by the client object (page 221, col 2) the method in the server object (page 222, col 1) wherein said step of calling invokes a method of the adapter (page 222, col 1, col 2; page 225, col 2), the method invoked in the adapter corresponding (i.e., *"The basic structure of object adapters is as a set of mappings which map method names from those presented to users to those used within the*

object interface.” The preceding text excerpts clearly indicate that the adapter maps/converts between the adapter interface and the actual server object interface. Therefore, a client’s invocation of an adapter method actually invokes a mapped server object method.) (page 222, col 2) to the method called in the server object (page 222, col 1); calling, by the method invoked in the adapter (page 222, col 1, col 2; page 225, col 2), a method in the object reference that corresponds (page 222, col 2) to the method invoked in the adapter (page 222, col 1, col 2; page 225, col 2); and calling, by the object reference, the method in the server object (page 222, col 1), wherein the adapter causes the object reference to invoke the method in the server object transparently to the client object such that the client object is unaware of the operation of the object reference (i.e., transparency of the target object reference method invocation is by definition of the adapter. This can be found in the design patterns book of Gamma, one of the inventors of the design patterns.) (page 222, col 1, col 2; page 225, col 2).

Office Action dated March 24, 2005, pages 2-5.

A prior art reference anticipates the claimed invention under 35 U.S.C. § 102 only if every element of a claimed invention is identically shown in that single reference, arranged as they are in the claims. *In re bond*, 910 F.2d 831, 832, 15 U.S.P.Q.2d 1566, 1567 (Fed Cir. 1990). All limitations of the claimed invention must be considered when determining patentability. *In re Lowry*, 32 F.3d 1579, 1582, 21 U.S.P.Q.2d 1031, 1034 (Fed Cir. 1994). Anticipation focuses on whether a claim reads on the product or process a prior art reference discloses, not on what the reference broadly teaches. *Kalman v. Kimberly-Clark Corp.*, 713 F.2d 760, 218 U.S.P.Q. 781 (Fed. Cir. 1983). Applicant respectfully submits that Trevor does not teach every element of the claimed invention arranged as they are in claims 1, 4, 12, 15, 23 and 24 of the present invention.

Amended independent claim 1, which is representative of amended claims 4, 12, 23 and 24 with regard to similarly recited subject matter, now recites:

1. A process for invoking a method of a server object in a distributed application in a distributed data processing system, the process comprising the computer-implemented steps of:
 - executing a client object in a client that implements a first programming environment, said client object being written for said first programming environment;
 - executing a server object in a server that implements a second programming environment that is different from said first programming

environment, said server object being written for said first programming environment;

executing said client object which begins an attempt to invoke a method in the server object;

in response to the client object beginning the attempt to invoke the method in the server object:

obtaining an object reference on the client for said second programming environment;

wrapping the object reference in an adapter which isolates the object reference from said client object, and the adapter performing data conversion between the object reference and the client object for different types of data, wherein the step of performing data conversion between the object reference and the client object for different types of data comprises:

responsive to receiving a method called by the client object,

parsing the method for an argument;

performing data conversion on the argument if necessary;

determining if the argument is wrapped by an adapter;

if the argument is wrapped by an adapter, unwrapping the adapter from the argument to retrieve an object reference of the argument, wherein the object reference of the argument provides a reference to a server object in the second programming environment representing the argument;

delegating the method called by the client object to the object reference of the argument;

responsive to detecting a return value, determining if the return value is an object reference for the second programming environment;

if the return value is an object reference for the second programming environment, wrapping the object reference with a suitable adapter based on a type of the object reference; and

returning the wrapped object reference to the client object.

(Emphasis added).

Trevor does not teach the features emphasized above. As discussed in the Abstract, Trevor presents a technique of realizing a shared object service by augmenting existing object facilities to provide management of their cooperative use. These facilities are realized through object adapters that provide additional cooperative facilities and greater control over the supporting infrastructure. Although Trevor teaches the concept of object adapters, Trevor does not teach that the object adapter performs data conversion between an object reference and a client object for different types of data in a manner recited in amended claims 1, 4, 12, 15, 23 and 24 of the present invention.

The Office Action alleges that Trevor teaches these features on page 222, column 2, where Trevor teaches that the basic structure of object adapters is as a set of mappings

which map method names from those presented to the users to those used within the object interface. Thus, Trevor merely teaches that object adapters have a set of mappings that map method names of a client request to method names of an object interface, such that when a client invokes a method on the object adapter, the object adapter calls a corresponding method on the object interface based on the set of mappings. Trevor fails to mention anything about performing data conversion between the client request and the object interface for different types of data.

To the contrary, Trevor teaches object adapters that are similar to the prior art approach as described on page 33, lines 10-20 of the current specification, which simply passes the arguments for the business method of the remote object when the business method is invoked from the adapter to the proxy object with no data conversion performed for the arguments. Trevor's object adapters are different from the adapter of the present invention in that Trevor's object adapters merely map method names of the users or clients to method names used within the object interface. There is no conversion of data for different types of data performed by the object adapters of Trevor.

On the other hand, the adapter of the present invention performs the conversion of data between the object reference and the client object. Instead of mapping the method name of the client object to a method name of the object reference, the adapter of the present invention parses the argument of the method invoked by the client object responsive to receiving the method called by the client object, determines if the argument is wrapped by an adapter, unwraps the adapter from the argument to retrieve an object reference of the argument if the argument is wrapped, delegates the method called by the client object to the object reference of the argument. In addition, the adapter of the present invention determines if a return value is an object reference responsive to detecting the return value, wraps the object reference with a suitable adapter based on a type of the object reference, and returns the wrapped object reference to the client object. Trevor's object adapters fail to perform data conversion according to the limitations described above.

Therefore, not only does Trevor fail to teach performing data conversion between the object reference and the client object for different types of data, Trevor also fails to teach responsive to receiving a method called by the client object, parsing the method for

an argument; performing data conversion on the argument if necessary; determining if the argument is wrapped by an adapter; if the argument is wrapped by an adapter, unwrapping the adapter from the argument to retrieve an object reference of the argument, wherein the object reference of the argument provides a reference to a server object in the second programming environment representing the argument; delegating the method called by the client object to the object reference of the argument; responsive to detecting a return value, determining if the return value is an object reference for the second programming environment; if the return value is an object reference for the second programming environment, wrapping the object reference with a suitable adapter based on a type of the object reference; and returning the wrapped object reference to the client object, as recited in amended claims 1, 4, 12, 15, 23 and 24 of the present invention.

In view of the above, Applicant respectfully submits that Trevor does not teach each and every feature of claims 1, 4, 12, 15, 23, and 24. At least by virtue of their dependency on claims 1, 4, 12, and 15 respectively, Trevor does not teach the features of dependent claims 2-3, 7-11, 13-14, 18-22. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 1-4, 7-9, 11-15, and 18-24 under 35 U.S.C. § 102(a).

II. 35 U.S.C. § 103(a), Alleged Obviousness, Claims 5-6, 10 and 16-17

The Office Action rejects claims 5-6, 10 and 16-17 under 35 U.S.C. § 103(a) as being allegedly unpatentable over Trevor et al., "The Use of Adapters to Support Cooperative Sharing", 1994 ACM, pages 219-230 in view of Anne Thomas, "Enterprise JavaBeans Technology", Patricia Seybold Group, © 1998, pages 1-24. This rejection is respectfully traversed.

As discussed in arguments presented above, Trevor fails to teach performing data conversion between the object reference and the client object for different types of data or responsive to receiving a method called by the client object, parsing the method for an argument; performing data conversion on the argument if necessary; determining if the argument is wrapped by an adapter; if the argument is wrapped by an adapter, unwrapping the adapter from the argument to retrieve an object reference of the

argument, wherein the object reference of the argument provides a reference to a server object in the second programming environment representing the argument; delegating the method called by the client object to the object reference of the argument; responsive to detecting a return value, determining if the return value is an object reference for the second programming environment; if the return value is an object reference for the second programming environment, wrapping the object reference with a suitable adapter based on a type of the object reference; and returning the wrapped object reference to the client object, as recited in claims 1, 4, 12, 15, 23, and 24. Anne Thomas also fails to teach the features of claims 4 and 15, from which claims 5-6, 10, and 16-17 depend.

As discussed in the introduction on page 1 of the reference, Anne Thomas teaches a server component model that comprises server components that run in an application server. On page 15, Anne Thomas teaches an enterprise Java Bean container that manages the enterprise beans that are deployed within it. Client applications do not directly interact with the enterprise bean. Instead, client application interacts with enterprise bean through two wrapper interfaces that are generated by the container, the EJB home interface and EJB object interface. As the client invokes operations using the wrapper interfaces, the container intercepts each method call and inserts management services. However, neither the container nor the wrapper interface performs data conversion between the client object and the object reference for different types of data. There is no teaching or suggestion in Anne Thomas for converting data between the object interface and the client object.

On page 14, Anne Thomas teaches that the EJB container acts as a liaison between the client and the enterprise bean. The container generates an EJB home interface and an EJB object interface at deployment time for each enterprise bean. The EJB home interface is used to create, find, and remove enterprise bean instances, while the EJB object interface is interpreted by the EJB container to insert lifecycle, transaction, state, security, and persistent rules on all operations. Thus, Anne Thomas is not concerned with converting data between the client and the enterprise bean. Anne Thomas is only concerned with providing services that manages the lifecycle, state, transaction, security, and persistence of the enterprise bean. Nowhere in the reference does Anne Thomas teach or suggest converting data between the client and the enterprise

bean for different types of data. Therefore, Anne Thomas does not teach or suggest the feature of performing data conversion between the object reference and the client object for different types of data.

Since Anne Thomas does not teach the feature of performing data conversion between the object reference and the client object for different types of data, Anne Thomas does not and would not teach responsive to receiving a method called by the client object, parsing the method for an argument; performing data conversion on the argument if necessary; determining if the argument is wrapped by an adapter; if the argument is wrapped by an adapter, unwrapping the adapter from the argument to retrieve an object reference of the argument, wherein the object reference of the argument provides a reference to a server object in the second programming environment representing the argument; delegating the method called by the client object to the object reference of the argument; responsive to detecting a return value, determining if the return value is an object reference for the second programming environment; if the return value is an object reference for the second programming environment, wrapping the object reference with a suitable adapter based on a type of the object reference; and returning the wrapped object reference to the client object, as recited in the features of claims 4 and 15, from which claims 5-6, 10, and 16-17 depend.

In addition, a person of ordinary skill would not have been led to modify or combine the teachings of Trevor and Anne Thomas to reach present claimed invention, because neither Trevor nor Anne Thomas mentions anything about performing data conversion between the client object and the object reference for different types of data. There is no teaching or suggestion in either of the reference of performing data conversion between the client object and the object reference for different types of data, let alone performing data conversion in the manner recited in claims 4 and 15 of the present invention. Therefore, it would not have been obvious for a person of ordinary skill in the art to modify or combine the teachings of Trevor and Anne Thomas to reach the presently claimed invention.

In view of the above, Applicant respectfully submits that neither Trevor nor Anne Thomas teaches or suggests the features of claims 4 and 15. At least by virtue of their dependency on claims 4 and 15 respectively, neither Trevor nor Anne Thomas teaches or

suggests the features of dependent claims 5-6, 10, and 16-17. Accordingly, Applicant respectfully requests withdrawal of the rejection of claims 5-6, 10, and 16-17 under 35 U.S.C. § 103(a).

III. Conclusion

It is respectfully urged that the subject application is patentable over the prior art of record and is now in condition for allowance.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: May 31, 2005

Respectfully submitted,



Wing Yan Mok
Reg. No. 56,237
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Agent for Applicant